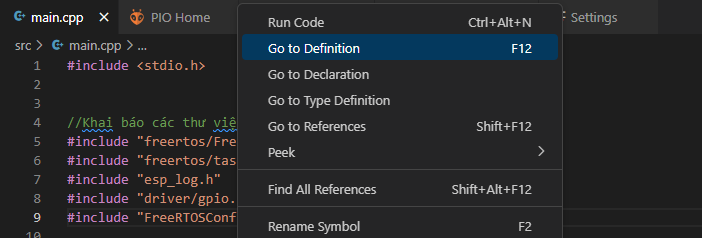
**9. Cấu hình chế độ độc quyền và không độc quyền**

9.1. Chế độ độc quyền (Non-Preemptive)

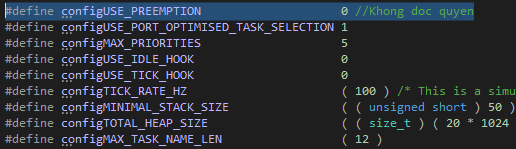
Để set chế độ độc quyền, ta nhấn vào “FreeRTOSConfig.h” và nhấn F12 hoặc chuột phải chọn Go to Definition



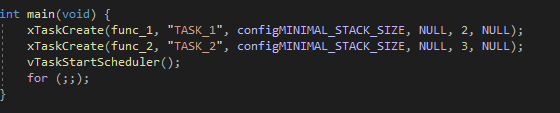
‘0’- Độc quyền: là độc chiếm CPU cho đến khi thực hiện xong hoặc rơi vào trạng thái blocked

‘1’- Không độc quyền: task đang chạy có thể bị chiếm quyền CPU nếu có task khác có độ ưu tiên cao hơn xuất hiện

Trong FreeRTOConfig.h thì ở dòng #define configUSE\_PREEMPTION thì cho = 0

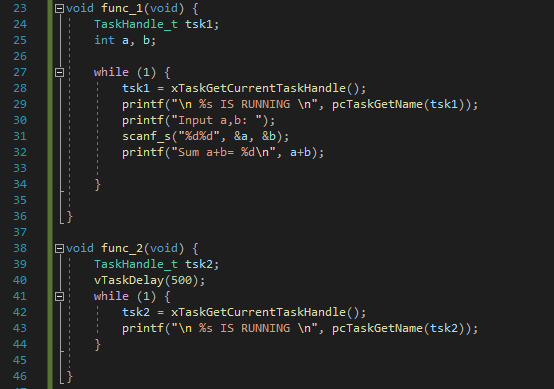


**Set up:** Sử dụng *chế độ độc quyền* với *độ ưu tiên* của Task 2 **lớn hơn** Task 1 **(Priority: 3 > 2)**



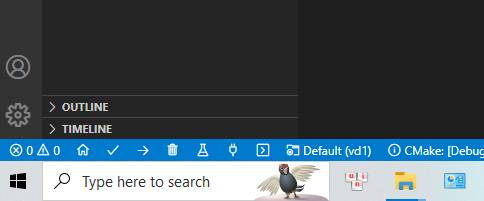
Ví dụ:

* TH1: Không có Task delay ở Task 1

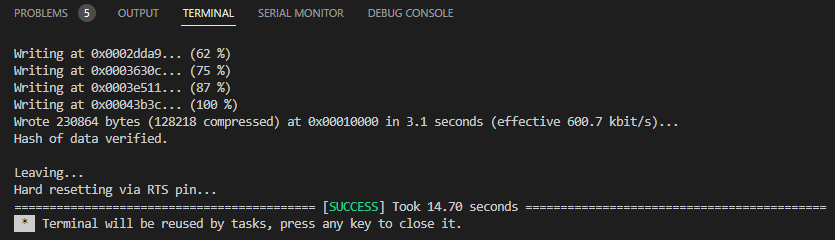


Giải thích: Task 2 có độ ưu tiên cao hơn task 1 (3>2). Do Task 2 set vào trạng thái ngủ(sleep) (vTaskDelay(500);). Do đó, Task 1 sẽ được thực thi trước, sẽ thực hiện phép cộng. Khi ở chế độ độc quyền, Task 1 không có TaskDelay, thì sẽ chạy vĩnh viễn cho tới khi kết thúc

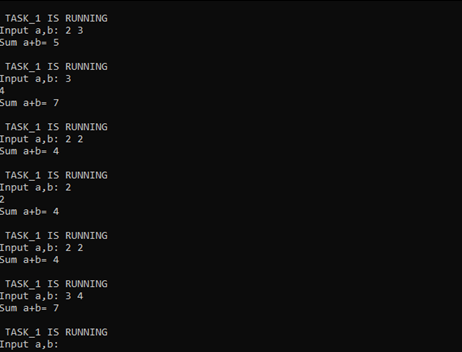
Sau đó sẽ ấn biểu tượng mũi tên bên góc trái dưới cùng màn hình để nạp code



Nếu thành công nó sẽ hiện:

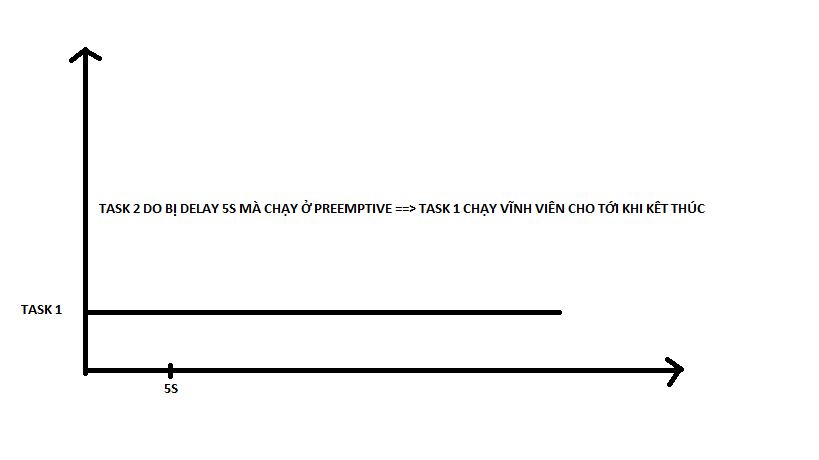


Sau đó quan sát màn hình terminal



Ở 5s đầu tiên, task 1 chạy và chạy vĩnh viễn cho đến khi kết thức tiến trình

Sơ đồ hoạt động:

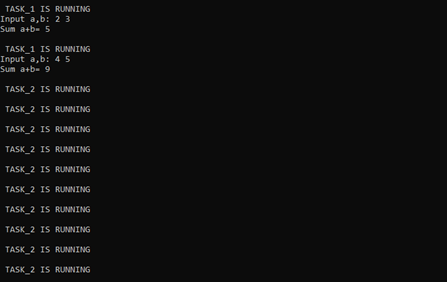


* TH2: Có Task delay ở while(1) trong Task 1



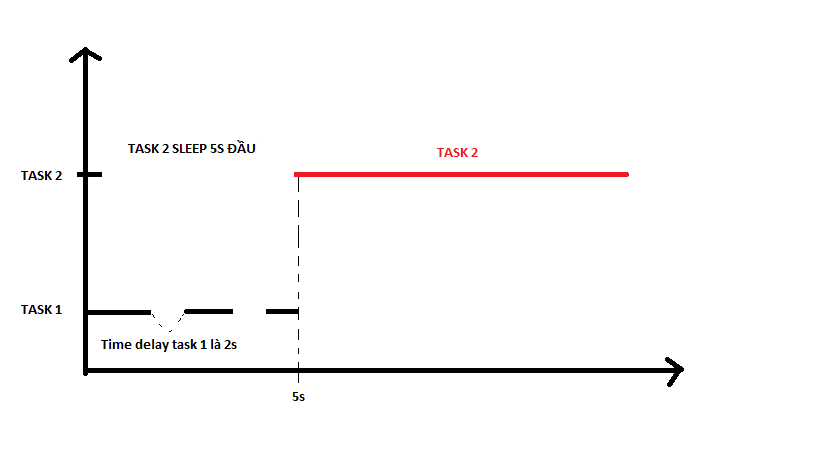
Giải thích: Task 2 có độ ưu tiên cao hơn task 1 (3>2). Do Task 2 set vào trạng thái ngủ(sleep) (vTaskDelay(500);). Do đó, Task 1 sẽ được thực thi trước, sẽ thực hiện phép cộng. Tuy nhiên, giữa các Task 1 có khoảng thởi gian sleep(Thởi gian ngủ giữa 2 Task 1 là 2s). Khi Task 1 rơi vào trạng thái sleep, Task 2 thức dậy, thì Task 2 sẽ chiếm quyền CPU vã sẽ chạy cho tới khi tiến trình kết thúc.

Sau đó chúng ta build và nạp code như Task 1

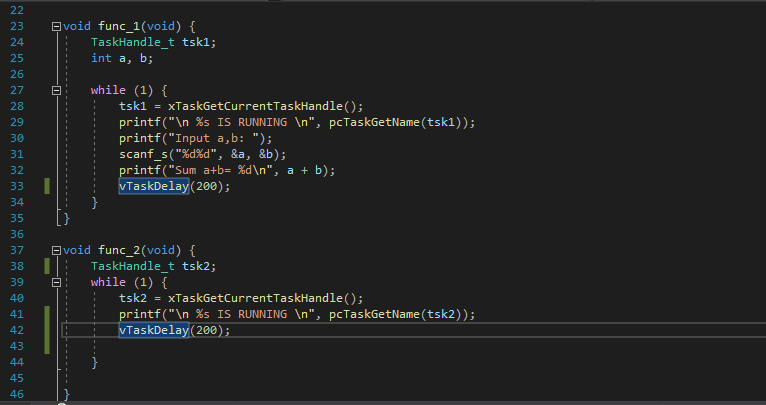


Task 1 sẽ chạy và khoảng thời gian sleep giữa 2 Task 1 là 2s, trong thời gian sleep, sau 5s, Task 2 tỉnh dậy và chiếm quyền CPU và chạy cho tới khi kết thúc

Sơ đồ hoạt động:

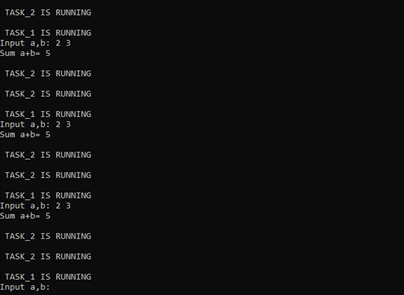


* TH3: Có Task delay ở while(1) trong cả Task 1 và Task 2

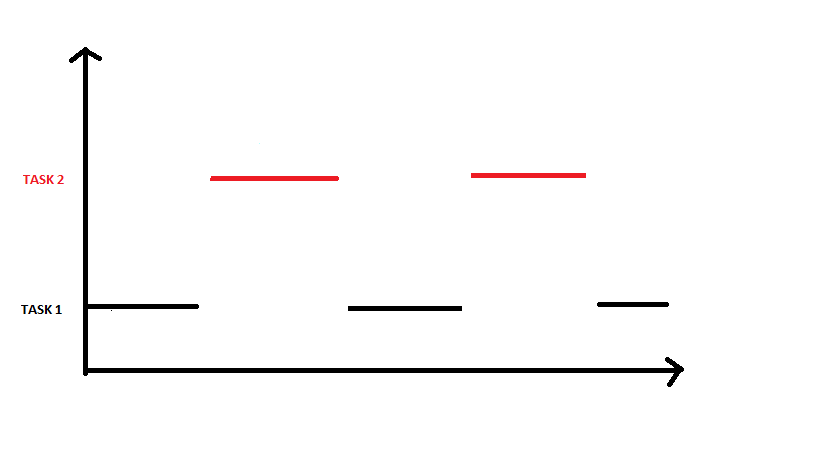


Giải thích: Task 2 có độ ưu tiên cao hơn Task 1, cho nên Task 2 sẽ được cập CPU để chạy, đến hàm TaskDelay thì Task 2 rơi vào trạng thái ngủ và Task 1 sẽ được cập CPU để chạy, và đến TaskDelay thì Task 1 rơi vào trạng thái ngủ và Task 1 được cấp CPU

Sau khi build và nạp code



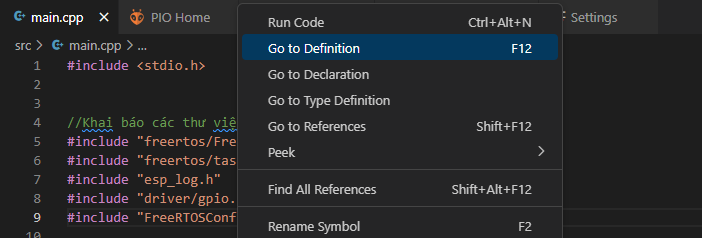
Sơ đồ hoạt động



🡪 Hệ điều hành sử dụng thuật toán dựa trên độ ưu tiên (Priority Based Algorithm) nghĩa là tiến trình nào có độ ưu tiên cao hơn sẽ được chạy trước và vì ở chế độ độc quyền nên khi có CPU thì tiến trình sẽ giữ cho đến khi hoàn thành hoặc khi tiến trình die/sleep/blocked.

9.2. Chế độ không độc quyền (Preemptive)

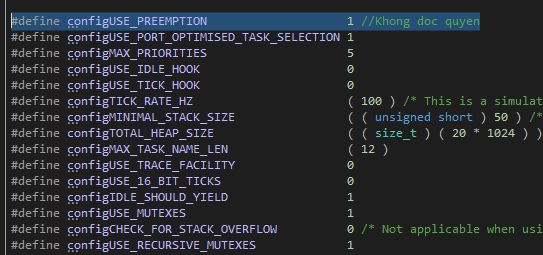
Để set chế độ độc quyền, ta nhấn vào “FreeRTOSConfig.h” và nhấn F12 hoặc chuột phải chọn Go to Definition



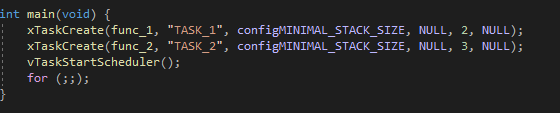
‘0’- Độc quyền: là độc chiếm CPU cho đến khi thực hiện xong hoặc rơi vào trạng thái blocked

‘1’- Không độc quyền: task đang chạy có thể bị chiếm quyền CPU nếu có task khác có độ ưu tiên cao hơn xuất hiện

Trong FreeRTOConfig.h thì ở dòng #define configUSE\_PREEMPTION thì cho = 1

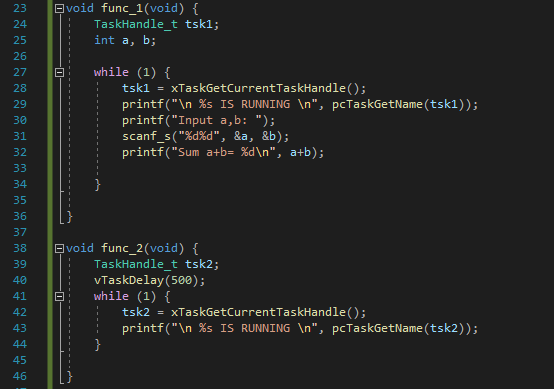


**Set up:** Sử dụng *chế độ không độc quyền* với *độ ưu tiên* của Task 2 **lớn hơn** Task 1 **(Priority: 3 > 2)**



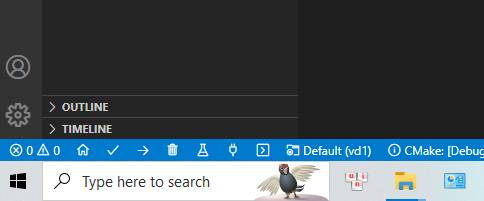
Ví dụ:

* Code mẫu

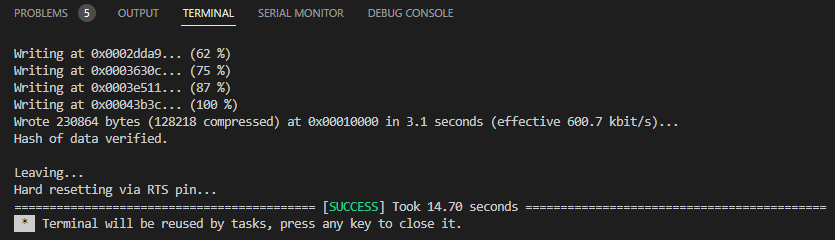


Giải thích: Task 2 có độ ưu tiên cao task 1 (3>2). Do Task 2 set vào trạng thái ngủ(sleep) (vTaskDelay(500);). Do đó, Task 1 sẽ được thực thi trước, sẽ thực hiện phép cộng. Sau 5s, thì tiến trình 2 thức dậy và sẽ được thực thi.

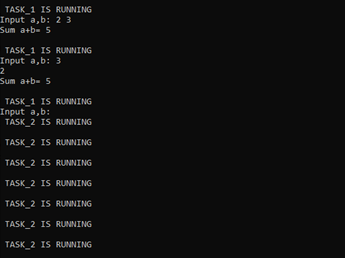
Sau đó sẽ ấn biểu tượng mũi tên bên góc trái dưới cùng màn hình để nạp code



Nếu thành công nó sẽ hiện:

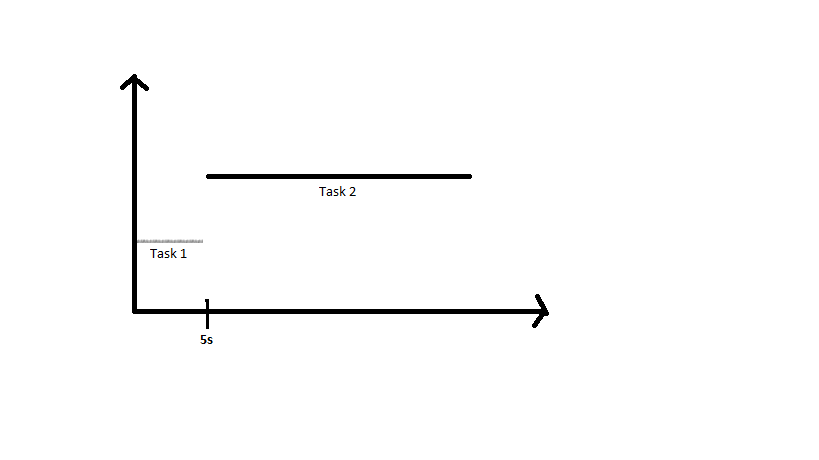


Sau đó quan sát màn hình terminal



Ở 5s đầu tiên, task 1 chạy và sau 5s task 2 sẽ được thực thi.

Sơ đồ hoạt động



* Hệ điều hành sử dụng thuật toán dựa trên độ ưu tiên (Priority Based Algorithm) và kết hợp với chiến lược phân phối xoay vòng (Round Robin với Quantumn = 1, TickRateHz = 10ms ) nghĩa là tiến trình nào có độ ưu tiên cao hơn sẽ được chạy trước còn khi có hai tiến trình có cùng độ ưu tiên chạy cùng lúc thì sẽ chạy theo chiến lược Round Robin.